CONNEXIONS

The Interoperability Report

July 1987 Volume 1, No. 3

ConneXions - The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 21370 Vai Avenue, Cupertino, CA 95014, USA
408-996-2042.

From the Editor

In this issue we bring you Part One of a two-part functional overview of the Internet protocols. This article, written by Barry Shein of Boston University, is a detailed look at the various components that make up the TCP/IP protocol family. Todays article is called "The Big Picture" examines how the various protocol elements fit together. Part Two is called "Anatomy of an Application Protocol" and gives an actual **C** program example. It will appear in our August issue.

Jon Postel and Bob Braden, of USC-ISI have been working on a very important RFC called "Requirements for Internet Gateways." Just as we went to press it was issued as RFC 1009. While it was in preparation it was referred to as RFC 985+ since it builds upon the work done by Dave Mills of the University of Delaware in RFC 985. Before RFC 1009 was issued we asked Bob Braden to give us an overview of this document.

As the use of the TCP/IP protocol suite in diverse environments increases, people are discovering all kinds of performance problems and a great deal of work is going into TCP "tuning." We feel that this research deserves attention and are planning a series of articles under the general heading "Improving your TCP." In this issue we bring you the first installment. Craig Partridge of BBN Laboratories looks at the infamous "TCP Timers." In future editions we plan to examine other aspects of performance improvement.

I have been attending the "core" Network Management Working Group meetings. The current status is that we are drafting a scope document which defines the issues to be addressed and gives some priorities and time schedules. This work will be presented at the next "plenary" which is being held in conjunction with the Internet Engineering Task Force meetng at Mitre, Washington on July 27th. We will keep you posted on the progress of the Working Group in future issues.

Advanced Computing Environments is continuing to sponsor conferences and tutorials on various topics related to computer communication. For more details, see "Upcoming Events," page 15.

The Internet Protocols -- A Functional Overview
Part One: The Big Picture

by Barry Shein, Boston University

Introduction

Users and system administrators starting with Internet protocols need what I would call a functional overview of the suite: what the pieces are, how they fit together, what the motivations were for the currently used protocols.

In general terms the Internet protocol suite is designed to provide various networking services within the framework of a packet switched network. This means that, at its lowest levels, data is sent as blocks (a *packet*) and higher-level protocol layers provide interpretation for these packets. Routing between networks and data integrity are provided transparently to a user's application by using standardized protocols.

One simple network which people encounter consists of a few systems on an Ethernet. Very often this is the first exposure systems administrators get to a network (other than hooking up their first system to the ARPAnet itself). Let's try a bottom-up view of such a network and see how the whole thing is put together. There are occasionally important exceptions to some of my descriptions; I will only attempt to describe the most common features. All programming examples are written in the **C** language and assume a machine with a 32-bit word and eight-bit bytes or *octets* as they are often called.

Ethernet overview

An Ethernet provides a method for transmitting data packets from a host to one or more hosts. These packets are defined by the Ethernet specifications [IEEE 802.3] to contain certain information in their headers followed by any arbitrary data, like an envelope with addresses on the outside and a message on the inside. These are defined as:

Destination Host Address	(48 bits)
Source Host Address	(48 bits)
Ethernet Packet Type	(16 bits)
...data...	

To a **C** programmer this might be defined as:

```
struct ether_header {
        unsigned char           destination[6];
        unsigned char           source[6];
        unsigned short          type;
};

struct ether_packet {
        struct ether_header     header;
        char                    data[];
};
```

The source host address corresponds to a unique number assigned by the manufacturer to the sending host's Ethernet board (my workstation, in hexadecimal, is 8:0:20:1:53:C3).

The destination address either corresponds to another host's Ethernet board or a "wildcard" address (called the Ethernet broadcast address, all bits on) which means the message is addressed to every system (Ethernet board) on this wire. The packet's data length is variable, up to 1500 octets.

Internet Protocol

Some of the data within an Ethernet packet corresponds to another packet, like an envelope within an envelope. The exact format of this message and its header depends upon the particular protocol being used. On an internetwork these are usually either IP (Internet Protocol), ARP (Address Resolution Protocol [RFC 826]), or RARP (Reverse ARP [RFC 903]) messages.

The primary purpose of the Internet Protocol (IP) is to provide low level routing for packets within an internetwork and to help provide an abstracted interface to any underlying hardware transport.

The structure of an IP packet header is:

Header Length	(4 bits)
Version Number	(4 bits)
Type of Service	(8 bits)
Total Length	(16 bits)
Identification	(16 bits)
Fragment Offset	(16 bits)
Time to Live	(8 bits)
Protocol	(8 bits)
Header Checksum	(16 bits)
Internet Source Address	(32 bits)
Internet Destination Address	(32 bits)

This is followed by a variable amount of data which the IP level does not interpret.

For the **C** programmer (the order of bitfields within an octet may need to be adjusted for a particular machine) this would be:

```
struct ip_header {
        unsigned char           header_length:4;
        unsigned char           version_number:4;
        unsigned char           type_of_service;
        unsigned short          total_length;
        unsigned short          identification;
        unsigned short          fragment_offset;
        unsigned char           time_to_live;
        unsigned char           protocol;
        unsigned short          checksum;
        unsigned int            source;
        unsigned int            destination;
};
```

Continuing from the Ethernet example, the entire packet would look like:

```
struct ip_packet {
        struct ether_header     ether_header;
        struct ip_header        ip_header;
        char                    data[];
};
```

continued on next page

The Internet Protocols -- A Functional Overview *(continued)*

Note that some of the data from the Ethernet packet is now being interpreted as the IP header, an envelope within an envelope. Details for this encapsulation are specified in [RFC 894.]

A full explanation of all the fields is described in [RFC 791] so I will only go over them briefly.

An IP header may contain optional extra information at the end (such as timestamp information) so the header can be variable in length. The first four bits gives the number of 32-bit words contained in the header. The "Type of Service" field is used to classify different packets at various priorities and security levels. The "Total Length" field gives the total number of octets including the data.

Fragmentation and reassembly

Data packets may be fragmented into pieces because of some limitation of the underlying hardware being used to transmit the data. The "Identification" field is used to group different fragments of the same logical packets for reassembly at the receiving end; the fragment offset indicates where in the final packet they should be placed (they might arrive out of order, or some may get lost and later need to be retransmitted, creating holes temporarily.) This automatic fragmentation protects an application program from needing to know what limitations its data may encounter en route to its destination. In a wide-area internetwork a packet may travel over several different transmission media, each with its own limitations.

The "Time to Live" field is a countdown value. Every time a packet passes through an intervening host (eg. from network to network), this value must be decremented by at least one. If it ever reaches zero the entire packet is discarded. This prevents packets from infinitely looping through a network due to some other problem.

The "Protocol" field simply indicates which higher-level protocol this IP packet is carrying (TCP, UDP, etc.) The "Header Checksum" field is a number generated at the source to verify that the header arrived uncorrupted. (No check on the data, other than its total length, is done at this level of the protocol.) The "Source" and "Destination" fields are the respective Internet addresses for the two communicating hosts or possibly an Internet (not Ethernet) broadcast address of all ones.

Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) is used to send messages between gateways. They use the IP header format and contain information such as a destination becoming unreachable or buffer space unavailability. It is described in [RFC 792.]

Higher level protocols

Two commonly used higher-level protocols are UDP (User Datagram Protocol [RFC 768]) and TCP (Transmission Control Protocol [RFC 793]). Each has a different application goal.

Packets containing data being sent via either of these protocols are wrapped in IP packets.

The simplest is the UDP packet which consists of:

Source port	(16 bits)
Destination port	(16 bits)
Length	(16 bits)
Checksum	(16 bits)

To the **C** programmer this would be:

```
struct udp_header {
        unsigned short              source_port;
        unsigned short              destination_port;
        unsigned short              length;
        unsigned short              checksum;
};
```

Following our example again from the Ethernet level, a UDP packet looks like:

```
struct udp_packet {
        struct ether_header         ether_header;
        struct ip_header            ip_header;
        struct udp_header           udp_header;
        char                        data[];
};
```

The "Source" and "Destination" ports are unique numbers used by the systems to match up data packets with particular programs' data channels (*sockets* in internet lingo). The "Length" is the number of octets in the message and the "Checksum" is an optional number generated by the sender which can be used to check that the data arrived without error.

TCP versus UDP

A TCP packet is similar but contains more fields to identify its sequence in a stream, help control the rate of data flow, and other provide information.

The main difference between the two protocols is that UDP is not guaranteed to be a reliable method of transmitting data while TCP does guarantee reliable data delivery. UDP is generally used where either reliability is not a major issue or the programmer would like to provide specially tailored reliability algorithms in an application.

TCP either delivers its data perfectly to an application program or the connection is (eventually) considered closed. UDP imposes less computational overhead. It's all a matter of application design trade-offs.

TCP is connection oriented; a virtual circuit is established between two applications and maintained until the channel is closed, similar to a phone call. UDP is connectionless; packets may be sent between two applications without maintaining a circuit, like sending letters through the mail.

continued on next page

The Internet Protocols -- A Functional Overview *(continued)*

Another important difference is that TCP provides a stream of octets with no other structure imposed (you can read in single octets or larger multiples) while UDP is packet oriented. RDP (Reliable Data Protocol [RFC 908]) is also becoming available to further round out this level of the protocol suite.

Addressing

Within an internet all hosts are assigned a unique 32-bit address which is independent of the underlying addressing scheme (that is, in our example, the Ethernet board address). These addresses come in four formats known as Class A, B, C and D. The first few bits in the first octet identifies which type of address is being used.

Class A addresses are indicated by the first bit being zero.The next 7 bits in the octet identify which network the host is on, and the remaining 24 bits identify the specific host within the network.

A Class B address is identified by the first two bits having the binary value 10; and uses the next 14 bits (the six left over in the first octet plus the entire following octet) to identify the network and the remaining 16 bits to identify the host.

A Class C address has its first three bits set to (binary) 110 and uses the top 21 bits to identify a network and the remaining 8 to identify the host.

The Class D address format is a recent addition which supports multi-casting (sending to a specific group of hosts, somewhere between a broad- cast to everyone and sending to one host).

Class A networks can contain up to just over 16 million hosts but there can only exist 128 Class A networks on one internetwork. At the opposite extreme there can be about 2 million Class C networks but each can only have slightly less than 255 hosts (zero is not available as a host's number). The ARPAnet is a Class A network, a small organization might contain a few Class C networks, a larger one might be serviced adequately by one or two Class B networks. Further details on the address formats are available in [RFC 997.]

These addresses are generally written in what is called dotted decimal notation. These are four numbers separated by periods, each number corresponds to one octet:

`10.4.0.44`	Class A network address
`128.197.0.1`	Class B network address
`192.12.185.1`	Class C network address

Naming

People like to assign names to their computers rather than type in the numbers all the time. In its simplest form this is accomplished by host tables which simply lists host's names and their corresponding numeric addresses so software can look this up for the users.

An entry in an Internet host table looks like:

```
HOST:192.12.185.1:BU-CS.BU.EDU:SUN:UNIX:TCP/TELNET,TCP/FTP,TCP/SMTP
```

UNIX hosts use a slightly different format:

```
192.12.185.1          bu-cs.bu.edu
```

A host can have more than one name. If a host has more than one address it implies that it has more than one hardware interface and is probably acting as a gateway between two networks.

Domains

The host table is becoming a thing of the past. It is too hard to maintain a complete table of hosts for a large network centrally. Name servers are being used which distribute the responsibility of maintaining name to numeric address pairs. They also provide new services such as allowing a host to offer to accept electronic mail for another, perhaps not directly attached, host. This will eliminate the need for keeping complicated forwarding information at a sender's end.

The domainified name of one of our nodes here is:

BU-CS.BU.EDU

This is a three-part address: the specific machine (BU-CS), the organization it belongs to (BU), and the top level domain the organization is in (EDU). A name can have as few as two parts (e.g. PURDUE.EDU). An organization may wish to use a multi-part name to break down its internal network further (e.g. MC.LCS.MIT.EDU). There are only several top-level domains currently defined:

EDU	Educational
GOV	Government
MIL	Military
COM	Commercial
NET	Network Service Organization
ARPA	ARPAnet
ORG	Unspecified Organization

Several countries, such as 'UK' for United Kingdom, have also been assigned their own top-level domains. The full list is available from the Network Information Center.

Each part of the address can have its own server. To resolve an address one first queries a known root server, in this case to find the address of the EDU server. The EDU server would then identify the BU server and the BU server would finally provide the address for BU-CS.BU.EDU.

The client portion of this system is known as a resolver. Usually this resolver is invoked automatically by software via subroutine calls within libraries. For example, on a 4.3 UNIX system the subroutine *gethostbyname()* communicates with the system's resolver, which in turn retrieves the information either from its local cache or from a query of a name server via the network. The domain system is described in [RFC 882, RFC 883, RFC 920, RFC 973 and RFC 974.]

continued on next page

The Internet Protocols -- A Functional Overview *(continued)*

Address Resolution Protocol

Given an internetwork address for a host, how does one find its corresponding Ethernet address? There are two ways. The first is to simply keep a table of pairs in each host on the Ethernet. Although this method is wonderfully simple, it tends to put the burden of administration of this table on a person somewhere. When an Ethernet board is replaced by service or a new system is added to the network someone has to update this table. Another method is the ARP protocol. The originating host has the internet address so it broadcasts a message to the entire Ethernet asking that whoever corresponds to this address please reply with their Ethernet address.

Upon reception of such an Ethernet address (an ARP reply from the queried host) the originating host will generally cache it away for later use. It's like a teacher calling out names in a classroom and then jotting down who is sitting where on a piece of paper. If people move or something is misplaced, the name can always be called out again later.

One advantage of this method is that only a small table need be allocated to maintain these pairs; if the table is full the host can safely throw away pairs that have not been used recently, knowing they can always be retrieved again at a later date.

The Reverse ARP is generally used with workstations. The intention is that one or more service hosts on a network contain the internet addresses for a group of workstations. The workstations do not contain their own addresses. At start-up a workstation broadcasts a *"Who Am I?"* message identifying itself by its Ethernet board address. The server then responds with the appropriate internet level address information. This is particularly useful in starting up diskless workstations.

Application protocols

There are many application protocols defined for use within the internet, among them are:

FTP	File Transfer Protocol (TCP)
TFTP	Trivial File Transfer Protocol (UDP)
Telnet	Network Terminal (TCP)
SMTP	Simple Mail Transfer Protocol (TCP)
Finger	Info about users on another system (TCP)
Time	Get system time from another host

Although not yet officially defined for the ARPAnet there is great interest in Sun's Network File System (allows mounting of disks logically across a network) and MIT's X-Window system (remote and local manipulation of graphical systems). Many other vendors are providing services for their databases, graphics, mail, printing, etc. with protocols built upon the Internetwork protocol suite.

Next month we look at a programming example. Part Two is called "Anatomy of an application protocol."

References

The ARPAnet protocols are defined via a series of publicly available documents called Requests for Comment (RFC.) Here is a list of RFCs referenced in this article.

[RFC 768]
Postel, J., "User Datagram Protocol"

[RFC 791]
Postel, J., "Internet Protocol"

[RFC 792]
Postel, J., "Internet Control Message Protocol"

[RFC 793]
Postel, J., "Transmission Control Protocol"

[RFC 826]
Plummer, D., "An Ethernet Address Resolution Protocol"

[RFC 882]
Mockapetris, P., "Domain Names - Concepts and Facilities"

[RFC 883]
Mockapetris, P., "Domain Names - Implementation and Specification"

[RFC 894]
Hornig, C., "A Standard for the Transmission of IP Datagrams over Ethernet Networks"

[RFC 903]
Finlayson, R.; Mann, T.; Mogul, J.; Theimer, M.,"A Reverse Address Resolution Protocol"

[RFC 908]
Velten, D., Hinden, R., Sax, J, "Reliable Data Protocol"

[RFC 920]
Postel, J., Reynolds, J., "Domain Requirements"

[RFC 954]
Harrenstien, K., Stahl, M., Feinler, E.,"NICNAME/WHOIS"

[RFC 973]
Mockapetris, P., "Domain System Changes and Observations"

[RFC 974]
Partridge, C., "Mail Routing and the Domain System"

[RFC 997]
Reynolds, J., Postel, J. "Internet Numbers" (Note that this is updated frequently.)

There are many other RFCs which fill in the picture. Many of these, plus some background articles, are contained in the DDN Protocol Handbook available from the DDN Network Information Center at SRI International. Call the NIC at 1-800-235-3155 for more information.

The following describes the Ethernet itself:

"The Ethernet - A Local Area Network," Version 1.0, Digital Equipment Corporation, Intel Corporation, Xerox Corporation, September 1980.

BARRY SHEIN is Manager of Special Projects for the Distributed Systems Group at Boston University and a Lecturer and Ph.D candidate in their Department of Computer Science. Previously he spent several years at the Harvard School of Public Health working on medical systems applications on the UNIX operating system.

A Specification of Gateway Requirements -- At Last

by Bob Braden, USC Information Sciences Institute

Introduction

It will come as no surprise to the vendors reading this newsletter that official specifications for building TCP/IP products are, to put it nicely, sketchy. There is a scattered set of vague RFCs plus a few generic examples to work from, and that is about it. The best known generic example is provided by Berkeley UNIX, which has been widely incorporated into vendor products.

We should point out that many RFCs are vague on purpose. Researchers, when they write papers, want to focus on the big picture, and leave "unimportant" implementation details unspecified. They are trying to demonstrate feasibility, not build a better product that will make money. Researchers generally get neither funding or recognition for writing careful engineering specifications.

In any case, the research community that developed the Internet architecture and protocols is well aware that the current situation is intolerable. The vendors need specifications and standards so they can build the robust, interoperable TCP/IP products we all want.

The lack of a specification for Internet gateways (also known as "IP routers") became acutely apparent when NSF began to build a major extension onto the Internet system. As a result of this important national initiative, dozens of new sites are coming to vendors to buy IP gateways. Both the customers and the vendors will be served by a specification document that can be referenced in an RFQ.

To meet this need, an advisory group to NSF commissioned Dave Mills, well known Internet researcher, to put together a gateway specification. He wrote a draft, which was published in May 1986 as RFC 985 ("Requirements for Internet Gateways -- Draft"). For the past six months an extensive revision to Mills' draft has been in preparation, and it will shortly be published as an RFC entitled "Requirements for Internet Gateways." This new RFC has just been assigned the number 1009, but it has generally been known as "RFC 985+" (or, to UNIX fans, as "RFC 985++"; an extra plus sign makes them feel more secure). [*]

The final RFC 985+ incorporates comments and suggestions made by many Internet wizards (including some vendor wizards), for which we are grateful. In the rest of this article, I will try to give some idea of the scope and contents of RFC 985+.

[*] The authors of RFC 985+ are Jon Postel (alias the Number Czar, alias the Protocol Czar, alias the editor of the RFC series) and Bob Braden, both at USC Information Sciences Institute. This work was supported in part by NSF and by DARPA.

The Big Picture

RFC 985+ is intended to tell a vendor what functions need to be in a gateway, and to tell customers what to ask for in a gateway. It attempts to be:

- *Comprehensive* - The document gathers together all aspects of a gateway -- the protocols it speaks, all of the different network drivers it could support, the algorithms it uses for routing and forwarding, and provisions for operations and maintenance.
- *Conservative* - It describes the current Internet architecture and tries to clarify the intent of the architects; it deliberately tries to avoid inventing any new architecture. In the process, it must also fill in some gaps. It also tries to strip away a few unnecessary or ill-advised protocol extensions which have crept into particular implementations over the years.
- *Selective* - This document describes only gateways, not bridges or hosts. In the future we hope to write a companion specification for hosts connecting to the Internet.

As we will make clear later, RFC 985+ is far from being complete. There are specification gaps at the current architectural level, and the architecture itself will be evolving to meet the challenges created by success. We hope to be able to publish later revisions to RFC 985+, to include new material and to update material which has changed. We will welcome comments and criticisms of the RFC.

In preparing RFC 985+, we had several principles in mind. These were as follows:

(1) *Require implementation of all features*

Reseachers have a way of leaving unimplemented any protocol features for which they do not have an immediate, pressing need. As a result of this and of a "Chicken and Egg" argument between hosts and gateways, there are a number of features which are not universally implemented. Examples are the ICMP messages for Timestamp, Address Mask, Information, and Record Route. RFC 985+ indicates clearly that these are required features (in fact, they are generally trivial to implement).

(2) *Keep strict protocol layering*

The rules for handling local network and IP-level broadcasts have been a source of some controversy. We chose to take the protocol-layering-purist viewpoint, keeping IP-level processing strictly independent of local network processing.

continued on next page

Gateway Requirements -- At Last *(continued)*

(3) *Sharpen Host / Gateway Distinction*

Some host operating systems include embedded gateway functions. Some of these, like Berkeley UNIX, have been carefully designed to follow the rules for a gateway, and the ready availability of gateway code in Berkeley UNIX has been very useful in many local internet environments. However, there is often a conflict between the host role and the gateway role of a general-purpose host running embedded gateway code. In general, the distinction between host and gateways (i.e., the ES/IS distinction in ISO terminology) is important, and RFC 985+ tries to reinforce it.

What's missing?

In conclusion, I want to give the reader some idea of the major gaps in the gateway specifications in RFC 985+, and what steps are afoot to fill those gaps.

EGP

EGP is terribly important in the present Internet, yet a number of aspects of EGP are causing severe heartburn. RFC 985+ mentions the problems, but cannot give solutions. However, many of these issues will be addressed by an EGP revision, known as "EGP 2," which is under preparation by the Internet Engineering Task Force. Even EGP 2 will be inadequate for the complex Internet system of today. Generalizing EGP for arbitrary anonymous system topology is a research topic that is currently being pursued, especially by Dave Mills. Incorporation of his work into the real world of the vendors will probably be some distance in the future.

Gateway monitoring and control

RFC 985+ describes the functions required for gateway monitoring and control, but there is currently no standard protocol for accomplishing these functions. Fortunately, there are efforts underway, again under the Internet Engineering Task Force umbrella, to define such protocols.

Interior gateway protocols

The RFC speaks of the desirability of an "Open IGP" standard, which would allow gateways from different vendors to exchange routing information reliably. There is no single effort in this area, but several efforts to document existing protocols (RIP and SPF) are under way.

Conclusion

The gateway requirements document has just been published as RFC 1009. It represents our best effort at defining what a gateway needs to do, at the present stage of the Internet. Those who have read early copies have told us that it is a significant step forward. However, we are aware of its limitations, and we will welcome feedback from vendors, users, and network wizards which will help improve future revisions to this important document.

BOB BRADEN has spent nearly a lifetime in the computer field, starting with 0th generation computers in 1951. A year ago he came to ISI from UCLA, where he spent the preceding 18 years working for the computer center. He first became involved with the ARPANET in 1970, when his systems group made the UCLA 360/91 an ARPANET host (1/1, now 10.1.0.1!). He participated in design of the ARPANET FTP and RJE protocols and in TCP development, and wrote the UCLA MVS TCP/IP code. He is a charter member of the IAB and is chairman of the End-to-End Protocols task force. Bob probably qualifies as an Internet Old Boy.

Improving Your TCP: Look at the Timers

by Craig Partridge, BBN Laboratories Inc.

Introduction

As IP networks have grown larger in the past few years many users and vendors have discovered that their TCP connections often fail or give very bad performance. The problem isn't TCP, but how TCP is often implemented. In this column, we look at one source of bad TCP performance: poor use of timers when estimating network round-trip times. There are other time-related issues, such as how to choose initial segment sequence numbers, but by far the most common timer problems in implementations are tied to estimating round-trip times.

One problem is that implementations often make unreasonable assumptions about how much time a packet may spend in transit. Two particular places to look in your code are:

- How long IP fragments are permitted to sit in the IP reassembly cache. Some implementations only store fragments for a few seconds. Typical round-trip times on the Internet are often in the tens of seconds, and fragments should be cached at least long enough that even if fragments get spaced out in transit, they can be reassembled at the destination.

- The upper bound set on the retransmission timeout. Most TCPs place an upper limit on how large the estimated round-trip time can be. The upper limit is often much too low; it should be *at least* the maximum possible round-trip time on the network.

Estimating round-trip times

Another common problem is that implementations often do a bad job of estimating the round-trip time. This is a serious problem since if the estimated round-trip time is too low, the implementation will retransmit too soon, putting needless extra packets on the network. If the estimated round-trip time is too high, then lost packets will not be retransmitted soon enough, causing sub-optimal throughput.

Dealing with samples

A frequent cause of poor round-trip time estimates is bad sampling of round-trip times. Many implementations sample the round-trip time only once per window of data. That's not often enough. Most round-trip time estimation systems include a certain amount of hysteresis to ensure that one anomalous sample doesn't badly skew the round-trip estimate. If an implementation samples only once per window, it may take several samples (and several windows of data) for the round-trip time estimates to adjust to changes in the network round-trip time. More frequent sampling keeps the estimated round-trip time more closely in tune with the network round-trip time.

continued on next page

Improving Your TCP: Look at the Timers *(continued)*

Another cause of poor estimates is accepting bad samples. Several recent papers have shown that the example retransmission timeout procedure in RFC 793 is sensitive to round-trip time samples which are the result of retransmissions, and that most implementations don't deal with this problem well. In particular, Zhang's work showed that the two most common methods for dealing with retransmissions don't work. In the past few months, Karn has suggested an algorithm which keeps proper time with the existing TCP algorithm. A detailed analysis of Karn's algorithm is due out soon. More generally, finding better retransmission timeout procedures is the subject of very active research.

Retransmission timeouts

Finally, it is possible to simply mistune the retransmission timeout procedure, by choosing poor control constants. This problem has been analyzed in some detail by Mills, and his writeup (RFC 889) is required reading for any TCP maintainer or implementor.

To conclude, if you are trying to improve the performance of your TCP implementation, take a good look at how it uses its timers.

References

Jain, Raj. "Divergence of Timeout Algorithms for Packet Retransmissions." Proceedings of the Fifth Annual International Phoenix Conference on Computers and Communications. 1986.

Mills, David. "Internet Delay Experiments," RFC 889. 1983.

Postel, Jon, ed. "Transmission Control Protocol," RFC 793. 1981.

Zhang, Lixia. "Why TCP Timers Don't Work Well." Proceedings of SIGCOMM 1986.

Partridge, Craig and Karn, Phil. "Improving Round-Trip Time Estimates in Reliable Transport Protocols," to be presented at SIGCOMM 1987.

CRAIG PARTRIDGE received his B.A. from Harvard University in 1983, and has been a part-time M.Sc. candidate there since 1986. For the past five years he has worked for BBN Laboratories on a variety of networking related projects including CSNET, the NSF Network Service Center (NNSC), and various projects concerned with distributed systems, IP transport protocols, and network management. In addition, he is a member of the Internet End-To-End Task Force, the Internet Engineering Task Force, and the Distributed Systems Architecture Board Task Force on Naming. He currently splits his time at BBN between CSNET, the NNSC, and managing a small TCP/IP networking project.

Upcoming Events

Advanced Computing Environments At Techmart Tutorial Schedule

The first of a series of Tutorial Sessions to be sponsored by Advanced Computing Environments At Techmart is scheduled for August 3-6, 1987 in Santa Clara, California. Each tutorial will include two full days of valuable information presented by the actual designers and developers of the systems each course covers. Insight on the latest trends and developments of the systems will provide attendees with training and knowledge that can be used and applied within their organizations. Advance registration is strongly recommended as space is limited.

August 3-4	*DARPA/NSF Internet System & Its Protocols* David Mills, University of Delaware
August 5-6	*Berkeley UNIX Networking* Mike Karels, UC Berkeley
August 5-6	*VAX VMS TCP/IP Networking* David Kashtan, SRI International

ISO Development Seminar

Advanced Computing Environments announces the ISO Development Seminar August 31 - September 2, 1987 in Monterey, California. Receive a thorough analysis of a completed implementation of the ISO protocol suite. Layer by layer explanation and evaluation. Comparison and details of discoveries plus transition strategies from TCP/IP to ISO. Speakers include leading computer scientists Lawrence Landweber, Marvin Solomon, Rob Hagens, Nancy Hall and Sue Lebeck from the University of Wisconsin*, who have worked for over two years on IBM-sponsored ISO implementation projects. Also speaking will be ISO Development Expert, Marshall Rose, from Northrop Research and Technology Center.*

*For identification purposes only.

For more information on these and other upcoming events, including program and registration, contact: Advanced Computing Environments: 408-996-2042.

UNIX is a trademark of AT&T Bell Laboratories.
VAX and VMS are trademarks of Digital Equipment Corporation.
IBM is a trademark of International Business Machines Corporation.
MS and Xenix are trademarks of Microsoft Corporation.

CONNEXIONS
21370 Vai Avenue
Cupertino, CA 95014

CONNEXIONS

PUBLISHER Daniel C. Lynch

EDITOR Ole J. Jacobsen

Subscribe to CONNEXIONS

U.S./Canada	$360. for 12 issues/year
University	$240. for 12 issues/year
International	$ 50. additional per year

Name ____________________ Title ____________________

Company ____

Address ____

City ____________________ State __________ Zip __________

Country ____________________ Telephone (   ) __________

□ Check enclosed (in U.S. dollars made payable to CONNEXIONS). □ Bill me/PO# __________

□ Charge my □ Visa □ Master Card Card # __________ Exp. Date __________

Signature ____

Please return this application with payment to: CONNEXIONS
21370 Vai Avenue
Cupertino, CA 95014